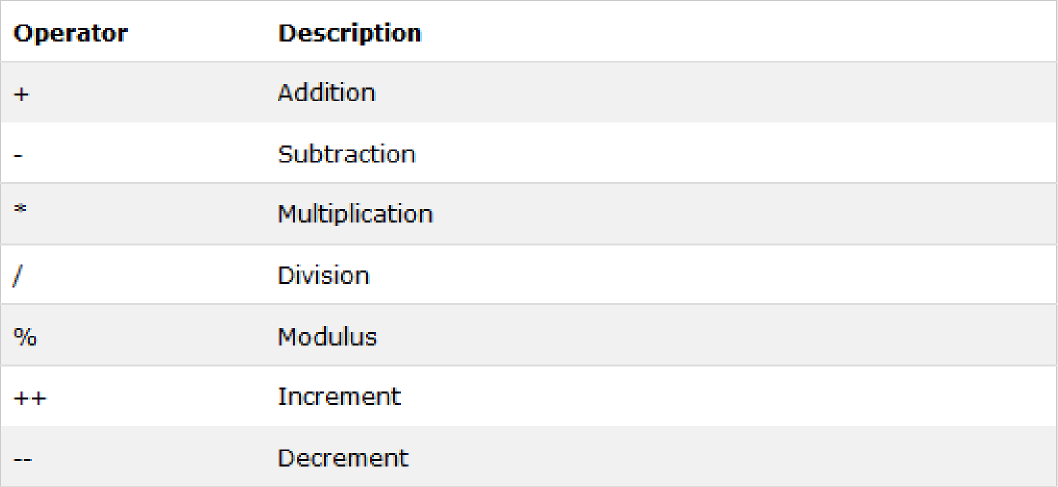
LẬP TRÌNH JAVASCRIPT DÀNH CHO NODEJS

1- Khai báo biến

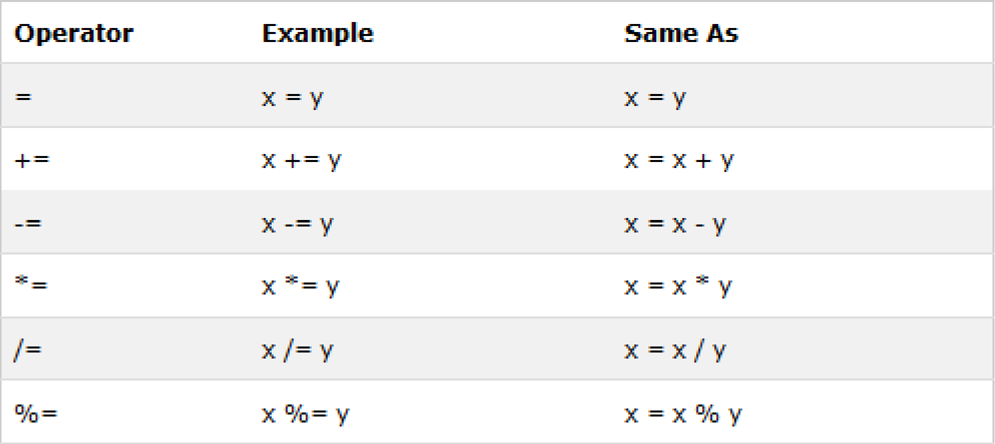
```
var x = 5;
var y = 6;
var z = x + y;
```

2- Toán tử



Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Cách khai báo toán tử khác:

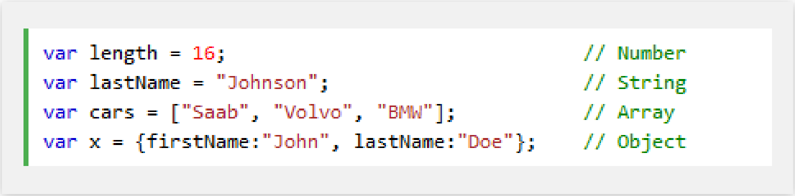


Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

3- Các phép toán so sánh

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

4 – Kiểu dữ liệu



```
var length = 16;                                  // Number
var lastName = "Johnson";                         // String
var cars = ["Saab", "Volvo", "BMW"];              // Array
var x = {firstName:"John", lastName:"Doe"};       // Object
```

5 – Functions (Hàm)

```
function myFunction(p1, p2) {
    return p1 * p2;              // The function returns the
product of p1 and p2
}
```

Ví dụ:

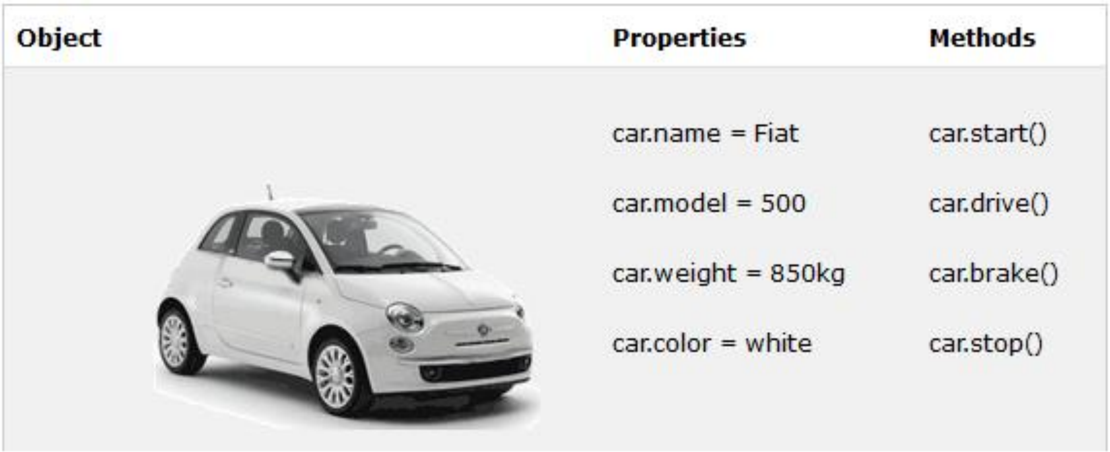
```
var x = myFunction(4, 3);        // Function is called, return value will end up in x

function myFunction(a, b) {
    return a * b;                // Function returns the product of a and b
}
```

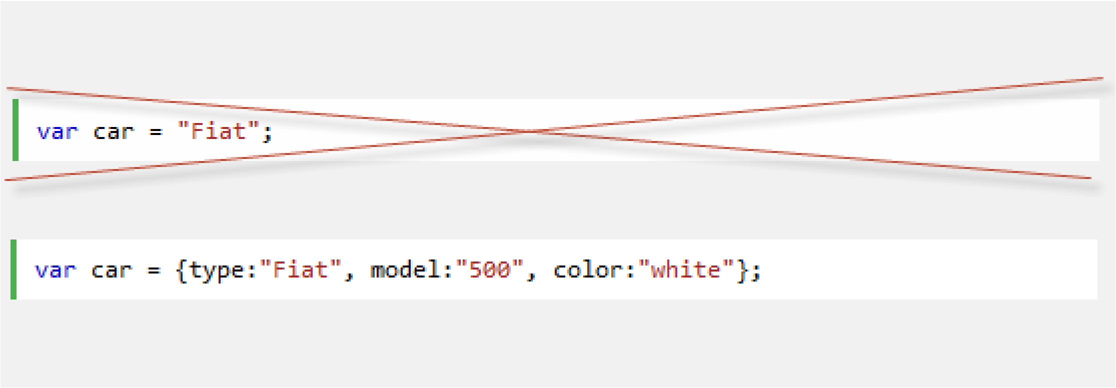
The result in x will be:

```
12
```

6- Objects



Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

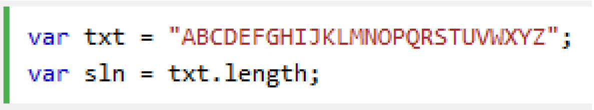


~~var car = "Fiat";~~

```
var car = {type:"Fiat", model:"500", color:"white"};
```

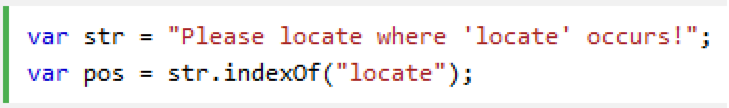
7- Các hàm xử lý Chuỗi

a- Đếm chiều dài chuỗi:

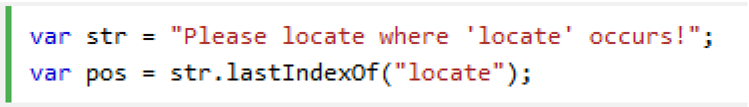


```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

b- Tìm kí tự trong chuỗi

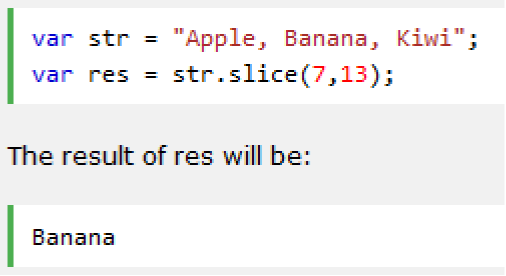


```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```



```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
```

c- Cắt chuỗi với Slice

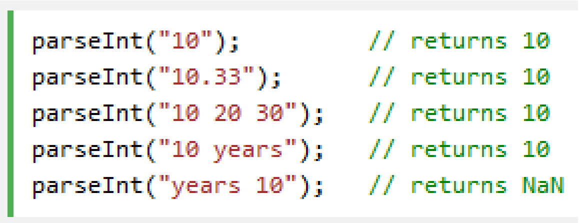


```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);
```

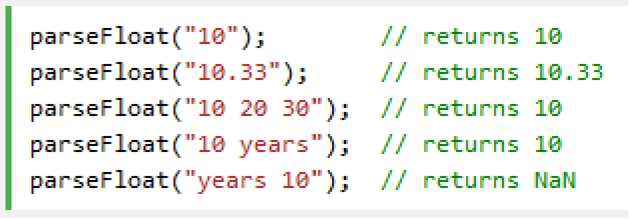
The result of res will be:

```
Banana
```

8- Ép kiểu dữ liệu

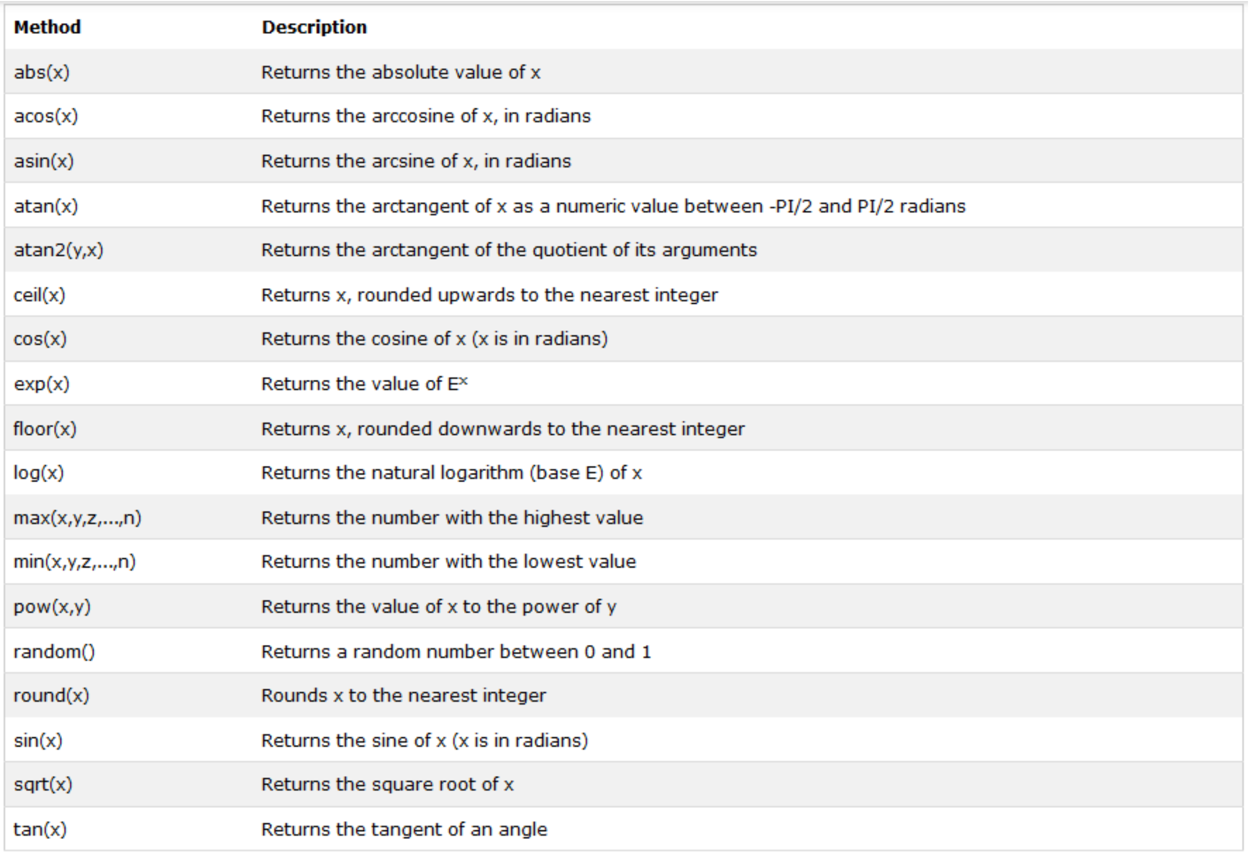


```
parseInt("10");         // returns 10
parseInt("10.33");      // returns 10
parseInt("10 20 30");   // returns 10
parseInt("10 years");   // returns 10
parseInt("years 10");   // returns NaN
```



```
parseFloat("10");         // returns 10
parseFloat("10.33");      // returns 10.33
parseFloat("10 20 30");   // returns 10
parseFloat("10 years");   // returns 10
parseFloat("years 10");   // returns NaN
```

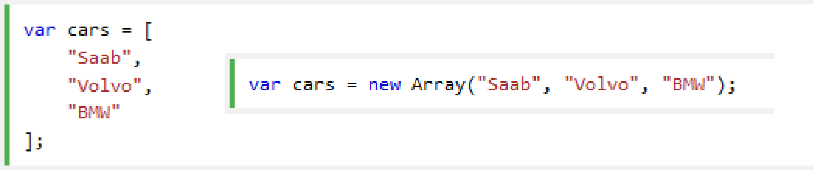
9 – Các hàm toán học



Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

10 – Mảng (Array)

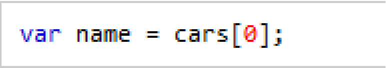
a- Tạo mảng



```
var cars = [
    "Saab",
    "Volvo",
    "BMW"
];
```

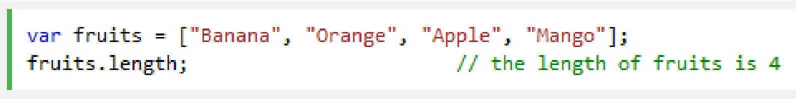
```
var cars = new Array("Saab", "Volvo", "BMW");
```

b- Truy cập phần tử



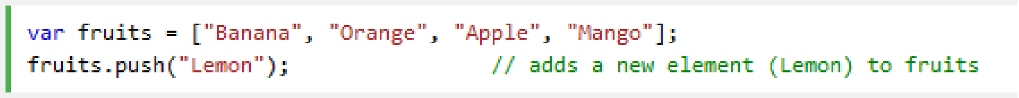
```
var name = cars[0];
```

C – Đếm số phần tử:



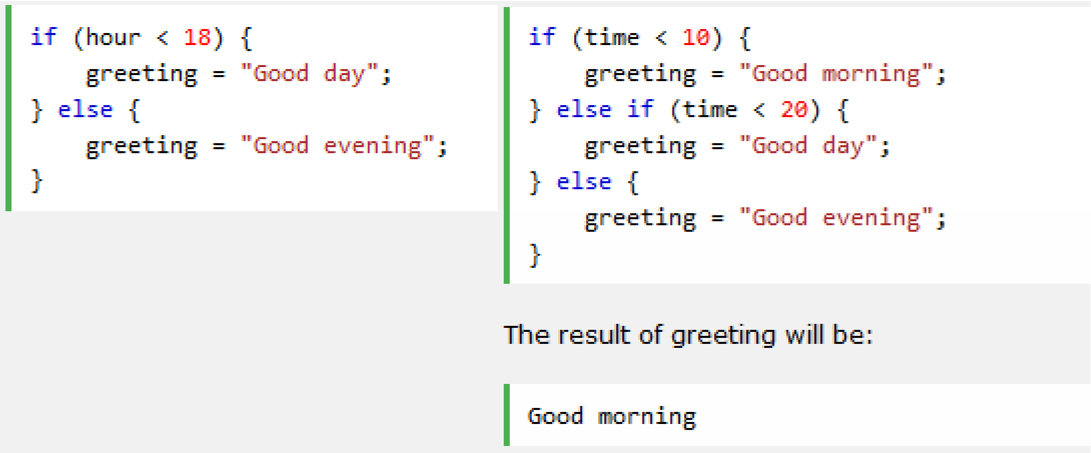
```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;                          // the length of fruits is 4
```

D – Thêm phần tử



```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Lemon");                // adds a new element (Lemon) to fruits
```

11- Câu lệnh If



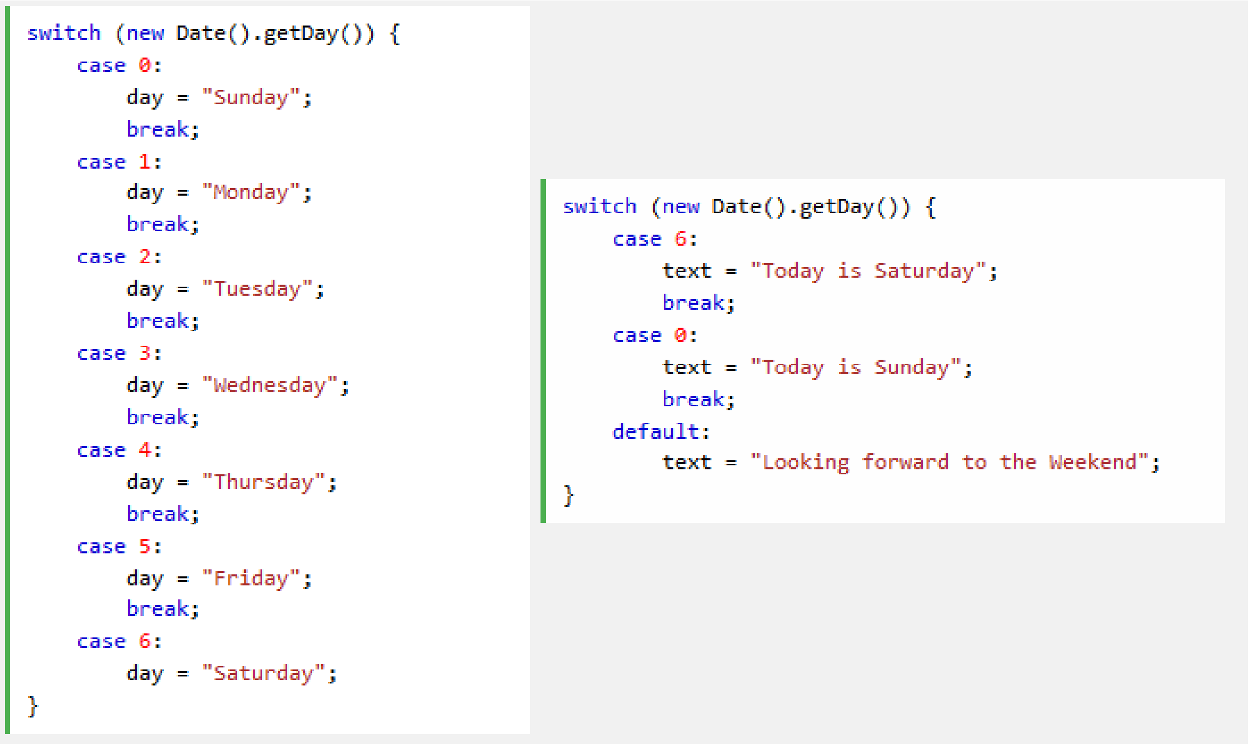
```
if (hour < 18) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

The result of greeting will be:

```
Good morning
```

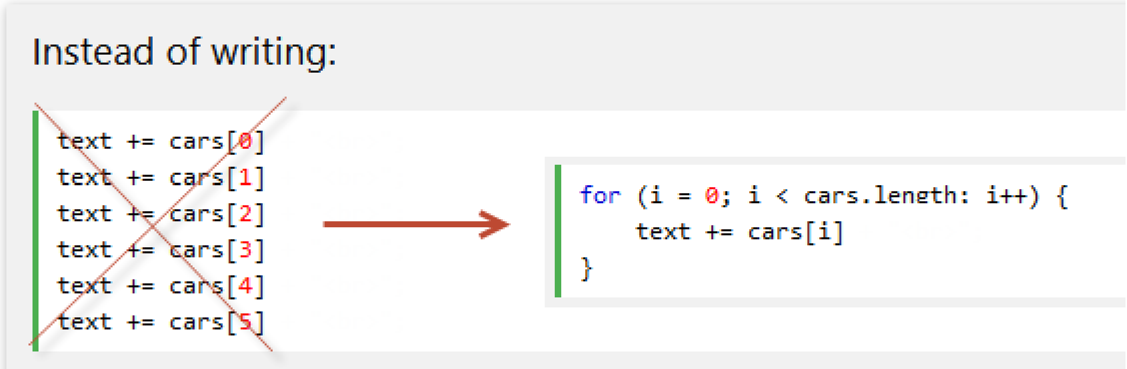
12 – Switch



```
switch (new Date().getDay()) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    case 4:
        day = "Thursday";
        break;
    case 5:
        day = "Friday";
        break;
    case 6:
        day = "Saturday";
}
```

```
switch (new Date().getDay()) {
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
        break;
    default:
        text = "Looking forward to the Weekend";
}
```

13- Vòng lặp For

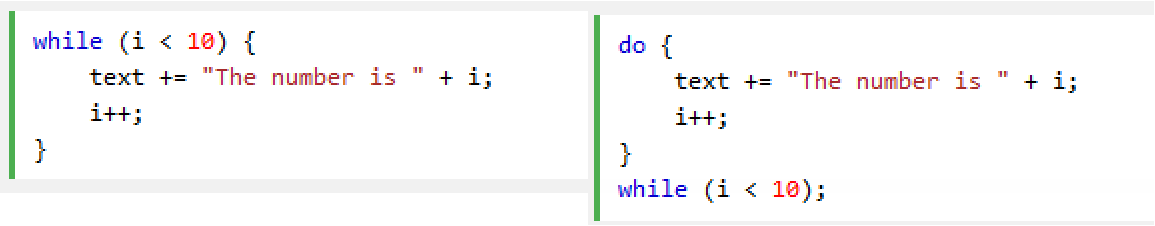


Instead of writing:

```
text += cars[0]
text += cars[1]
text += cars[2]
text += cars[3]
text += cars[4]
text += cars[5]
```

```
for (i = 0; i < cars.length; i++) {
    text += cars[i]
}
```

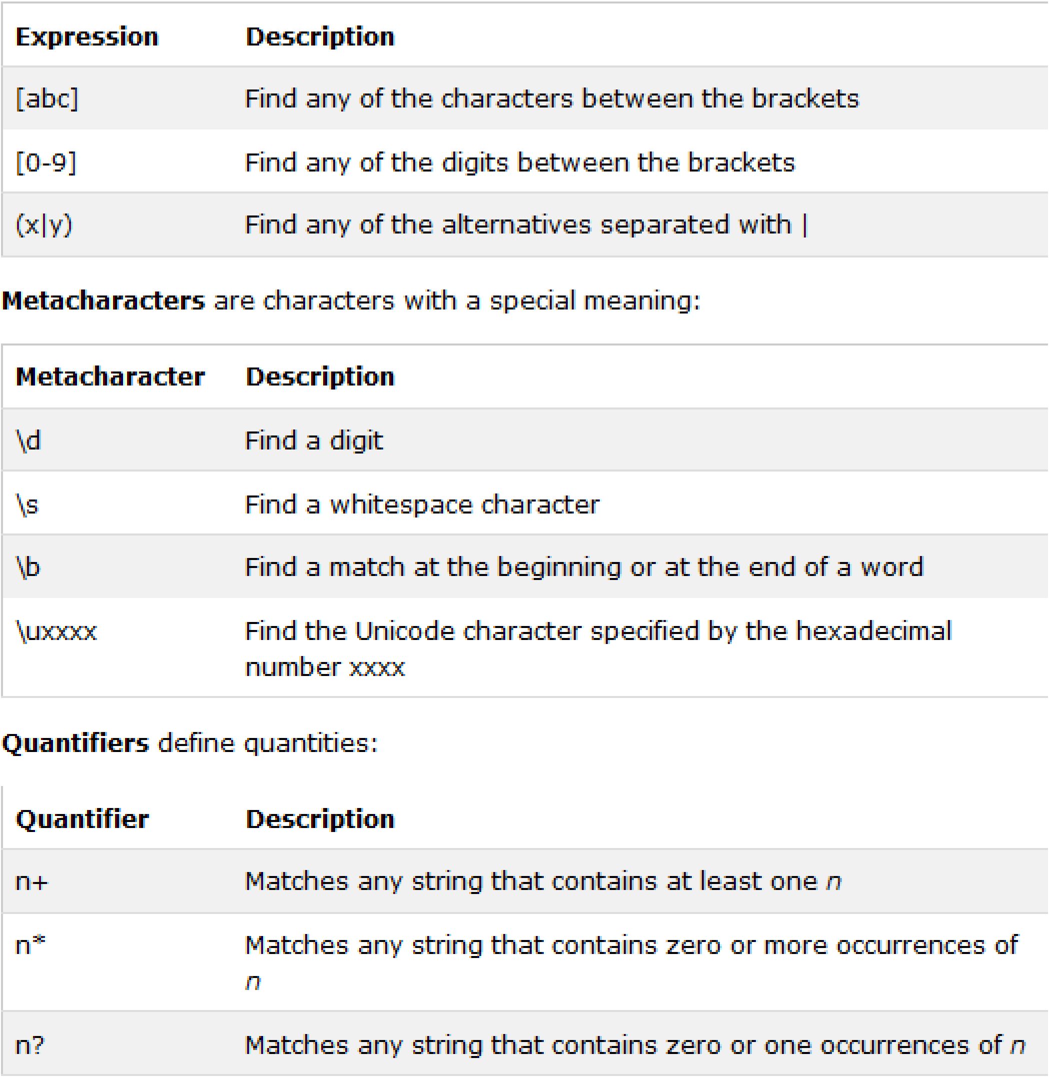
14- Vòng lặp While



```
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

15- Regular Expression



Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacters are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*